Administration des services

TP 1 - Dépannage réseau sous Linux et surveillance du système et des services

1.Introduction

Le but de ce TP était de revoir et pratiquer quelques commandes essentielles sous Linux pour le dépannage réseau et la surveillance des services.

Nous avons travaillé sur une VM Debian 13 clonée sur Proxmox, ce qui nous a permis de tester les commandes sans risque pour la machine originale.

Le TP est divisé en deux parties principales :

Dépannage réseau : vérifier et corriger la configuration IP, tester la connectivité et la résolution DNS.

Surveillance et gestion des services : lister les services actifs, visualiser les connexions et gérer le démarrage automatique des services.

2. Préambule : Clonage de la VM (DB13)

Tout d'abord avant de commencer le TP je me suis rendue sur proxmox à l'adresse suivante ( ) et j'ai cloner la vm de debian 13 en choissisant le serveur 3 (car il était peu utiliser à ce moment là) , j'ai ensuite choisis le VMID (1 + mon n°octet + deux chiffres) , j'ai donné un nom à cette vm (NomTP-OS-n°TP) je l'ai ensuite placé dans mon pool.

J'ai utilisé un clonage lié. Ce type de clonage est rapide à créer et peu gourmand en espace disque, car il partage certains fichiers avec la VM originale. Cependant la VM clonée reste dépendante de l'original.

Je n'ai pas eu bessoin de mettre à jour les sources listes car elle était déja bonne , mais j'ai mis a jour le system grace a la commande :

3. Dépannage réseau

Dans cette partie du TP, l'objectif était de **diagnostiquer et vérifier la configuration réseau** de notre client Linux local en utilisant plusieurs commandes essentielles.

3.1 Identification de la carte réseau

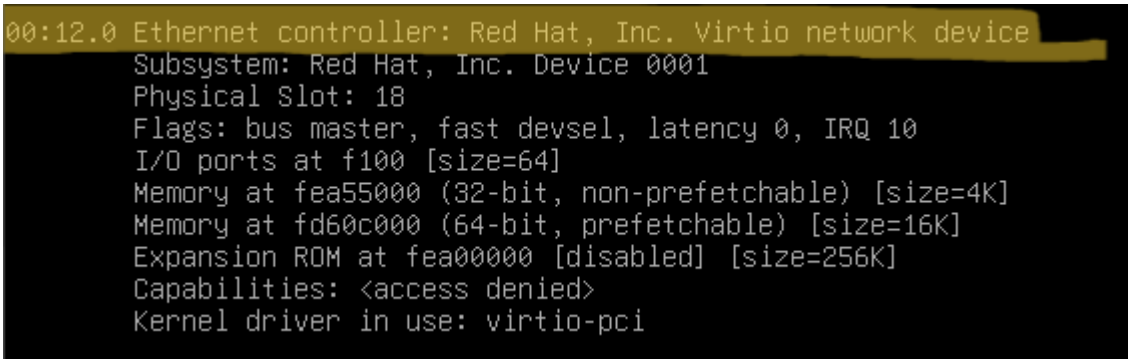
Pour commencer, j'ai utilisé la commande :

```
lspci -v
```

Que fait la commande lspci -v ?

- lspci liste tous les périphériques PCI (cartes réseau, cartes graphiques, contrôleurs USB, etc.)

- L'option -v nous donne plus de détails sur chaque périphérique : driver utilisé, état du périphérique…



La sortie m'a indiqué :

00:12.0 Ethernet controller: Red Hat, Inc. Virtio network device

Carte réseau : Virtio network device

Périphérique associé : 00:12.0

Cette étape permet de savoir quelle carte réseau est détectée par le système et sous quel périphérique PCI elle est reconnue, ce qui est indispensable pour le dépannage réseau.

Ensuite, j’ai consulté les messages du noyau relatifs à cette carte avec la commande :

dmesg | grep virtio

Au début, j’ai obtenu un message d’erreur : *“échec de lecture du tampon du noyau : opération non permise”*.
Après m’être renseigné, j’ai compris que je devais exécuter la commande avec les droits root. J’ai donc ajouté sudo devant la commande et elle a fonctionné correctement.

```
 1.924004] virtio_blk virtio2: 2/0/0 default/read/poll queues
 1.976442] virtio_blk virtio2: [vda] 67108864 512-byte logical blocks (34.4 GB/32.0 GiB)
 2.057582] virtio_net virtio3 ens18: renamed from eth0
```

Cette commande permet de vérifier comment l’interface réseau a été initialisée, quel driver est utilisé par le système et si des erreurs ou problèmes ont été détectés lors du démarrage. Elle aide à confirmer que la carte réseau fonctionne correctement dès le boot de la machine.

3.2 État du lien réseau

Pour vérifier que le lien réseau est actif, j’ai voulue utilisé la commande :

`mii-tool`

`Mais ça n’a pas fonctionnée , ça me renvoyé not found`

`J’ai donc trouvé une alternative à cette commande sur le WEB : ip link show`

```
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether bc:24:11:e4:56:0b brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    altname enxbc2411e4560b
$ _
```

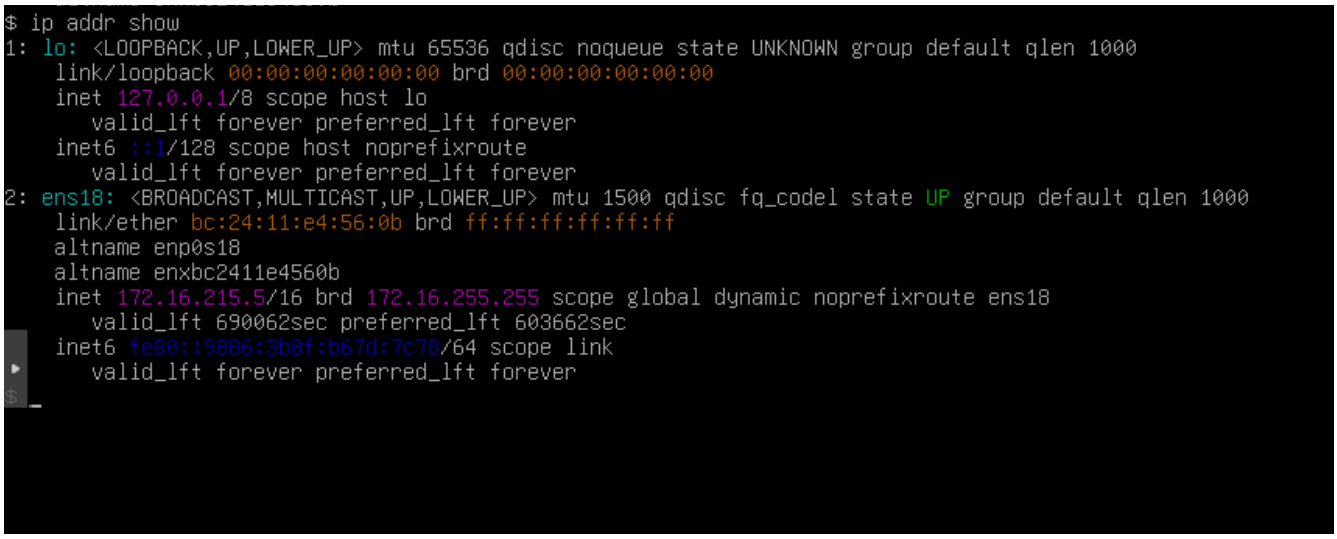
On peut bien voir que ma VM dispose de deux interfaces réseau : lo , c’est l’interface loopback , elle sert à communiquer avec la machine elle-même et elle n’est pas connectée au réseau externe.

Et après y’a ens18 , c’est mon interface principale connectée au réseau. La sortie de ip link show a indiqué que l’interface était **UP** et que le lien physique était actif (LOWER_UP), ce qui confirme que la VM peut communiquer sur le réseau.

3.3 Informations sur l’interface réseau

J’ai ensuite utilisé la commande :

```
ip addr show
```



Les informations principales obtenues sont les suivantes :

-Nom de l’interface : ens18

-Adresse IPv4 : 172.16.215.5/16

-Masque : 255.255.0.0

-Adresse de broadcast : 172.16.255.255

-Adresse MAC : fe80 : : 9886 : 3b8f : b67d : 7c78/64

L’indication que l’interface est opérationnelle se trouve sur la première ligne, avec la mention :

BROADCAST, MULTICAST, ,UP, LOWER_UP mtu 1500

3.4 Test de la pile TCP/IP

Pour vérifier que la pile TCP/IP fonctionne correctement, j'ai testé l'adresse loopback :

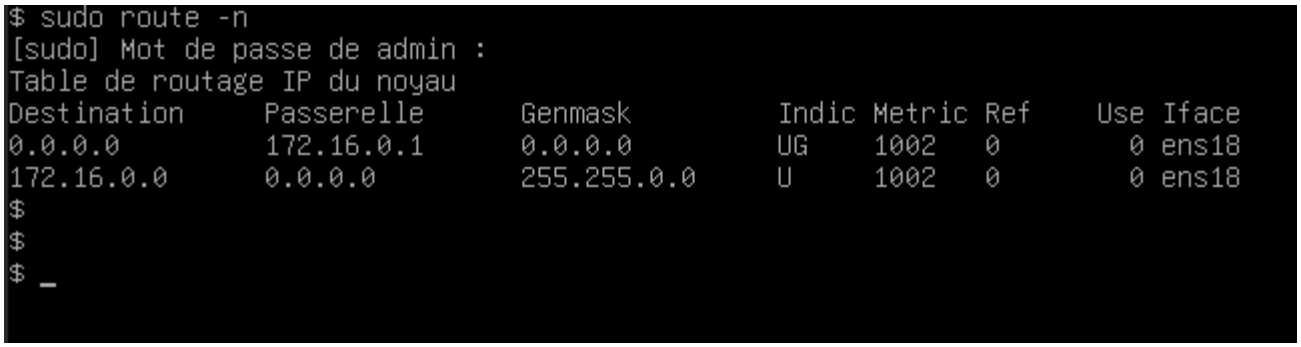
```
ping 127.0.0.1
```

Cette adresse correspond à la machine elle-même, donc aucun paquet ne quitte la VM.Le test a réussi, ce qui confirme que les liaisons TCP/IP locales sont correctement configurées.

3.5 Vérification de la passerelle

Pour connaître l'adresse de la passerelle, j'ai utilisé la commande :

```
route -n
```



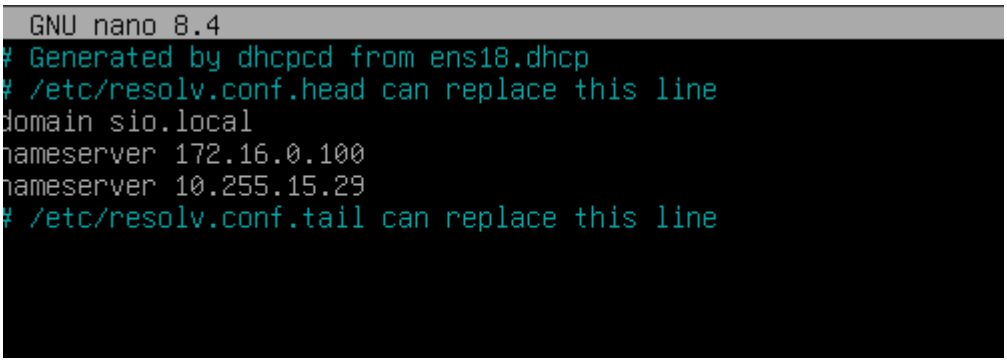
La passerelle est : 172.16.0.1

La passerelle permet à la VM de communiquer avec d'autres réseaux, notamment Internet.

3.6 Test de la résolution DNS

Pour tester la résolution des noms de domaine, j'ai commencé par vider le fichier `/etc/resolv.conf` :

```
nano /etc/resolv.conf
```



Cette action supprime la configuration du serveur DNS de la machine.Sans DNS, le système ne peut plus traduire un nom de domaine en adresse IP.

Ensuite, j'ai testé la connectivité :

```
ping www.google.fr
```

Résultat : unknown host www.google.fr → le nom de domaine ne peut pas être résolu.

Puis, j'ai testé directement l'adresse IP de Google :

```
ping 209.85.147.94
```

Résultat : la connexion fonctionne correctement.Cela montre que la connexion réseau fonctionne, mais le problème venait de l'absence de DNS.

J'ai ensuite remis les bonnes informations dans `/etc/resolv.conf` :

```
domain sio.local
```

```
nameserver 172.16.0.100
```

Après cette modification, la commande :

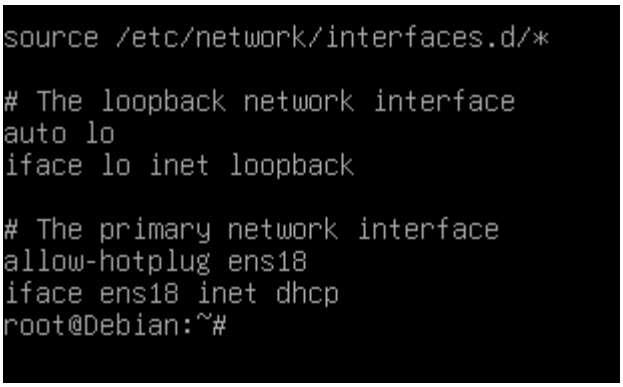
```
ping www.google.fr
```

fonctionne correctement, confirmant que la machine peut maintenant résoudre les noms de domaine.

3.7 Comparaison des fichiers de configuration

J’ai comparé les fichiers de configuration réseau sur mon client Linux, à savoir /etc/network/interfaces et /etc/resolv.conf.

Le fichier /etc/network/interfaces configure l’interface réseau principale, ici ens18. Sur mon client, cette interface est configurée en DHCP, ce qui signifie que l’adresse IP, le masque, la passerelle et l’adresse de broadcast sont attribués automatiquement par le serveur DHCP.



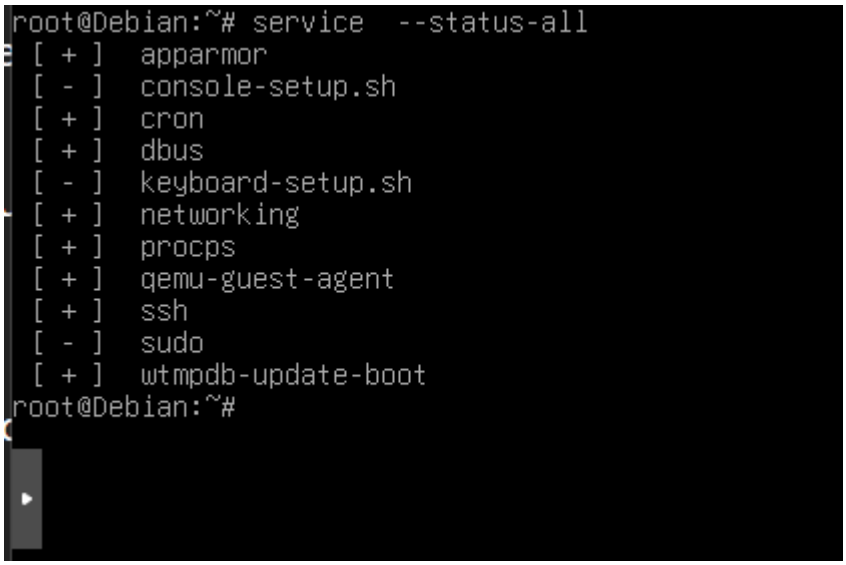
En revanche, le fichier /etc/resolv.conf est dédié à la configuration des serveurs DNS. Il contient le nom de domaine local et l’adresse du serveur DNS utilisé pour résoudre les noms de domaine. Ces informations sont définies manuellement et ne dépendent pas du DHCP.

La différence principale entre ces deux fichiers est donc que l’un configure l’adresse IP et l’interface réseau, tandis que l’autre configure le DNS. Les deux sont indispensables pour que la machine puisse communiquer correctement sur le réseau et accéder à Internet.

4.SURVEILLANCE ET GESTION DES SERVICES

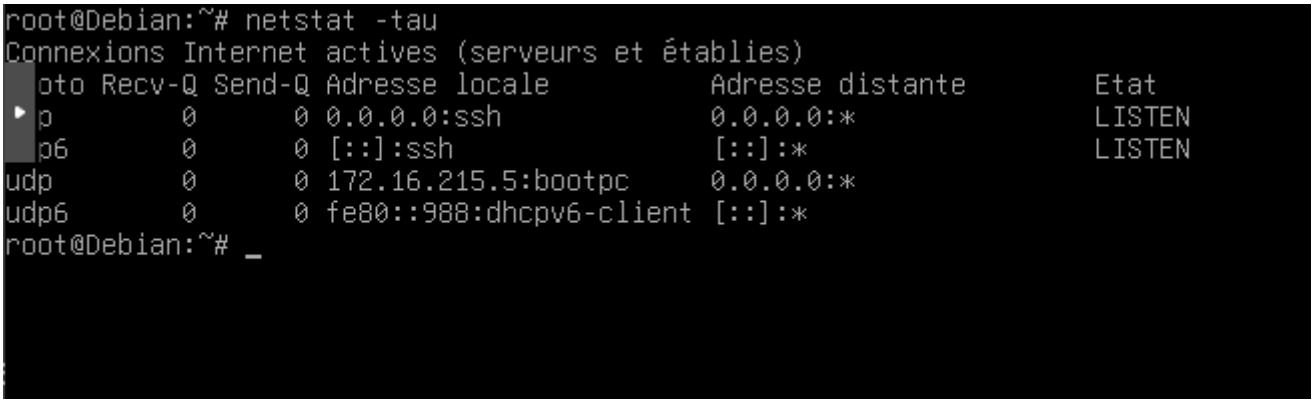
4.1Surveillance des services

Pour identifier les services offerts par mon serveur DB13 , j'ai utilisé la commande `service--status-all`. Cela permet de connaître tous les services actifs ou disponibles sur la machine, tels que le serveur SSH, le serveur web , ou encore le service de gestion du réseau.



Pour vérifier les connexions réseau en écoute et établies sur le serveur, j'ai utilisé la commande :

`netstat -tau`



Cette commande liste toutes les connexions TCP et UDP.

Elle permet de savoir quelles applications ou services écoutent sur quelles ports, et quelles connexions sont actuellement actives.

4.2.Gestion des services

Pour savoir quels services sont démarrés automatiquement au boot du serveur, j'ai utilisé :

```
systemctl list-unit-files --type=service | grep enabled
```

```
root@Debian:~# systemctl list-unit-files --type=service | grep enabled
apparmor.service                              enabled         enabled
console-setup.service                         enabled         enabled
cron.service                                  enabled         enabled
cryptdisks-early.service                      masked          enabled
cryptdisks.service                            masked          enabled
e2scrub_reap.service                          enabled         enabled
getty@.service                                enabled         enabled
grub-common.service                           enabled         enabled
hwclock.service                               masked          enabled
ifupdown-wait-online.service                  disabled        enabled
keyboard-setup.service                        enabled         enabled
networking.service                            enabled         enabled
nftables.service                              disabled        enabled
serial-getty@.service                         disabled        enabled
ssh.service                                   enabled         enabled
sshd-keygen.service                           enabled         enabled
sshd@.service                                 indirect        enabled
sudo.service                                  masked          enabled
systemd-confext.service                       disabled        enabled
systemd-fsck-root.service                     enabled-runtime enabled
systemd-network-generator.service             disabled        enabled
systemd-networkd-wait-online.service          disabled        enabled
systemd-networkd-wait-online@.service         disabled        enabled
systemd-networkd.service                      disabled        enabled
systemd-pcrlock-file-system.service           disabled        enabled
systemd-pcrlock-firmware-code.service         disabled        enabled
systemd-pcrlock-firmware-config.service       disabled        enabled
systemd-pcrlock-machine-id.service            disabled        enabled
systemd-pcrlock-make-policy.service           disabled        enabled
systemd-pcrlock-secureboot-authority.service  disabled        enabled
systemd-pcrlock-secureboot-policy.service     disabled        enabled
systemd-pstore.service                        enabled         enabled
systemd-remount-fs.service                    enabled-runtime enabled
systemd-sysext.service                        disabled        enabled
systemd-timesyncd.service                     enabled         enabled
systemd-udev-load-credentials.service         disabled        enabled
wtmpdb-update-boot.service                    enabled         enabled
x11-common.service                            masked          enabled
root@Debian:~#
```

Cette commande affiche tous les services configurés pour démarrer automatiquement à chaque démarrage de la machine.

- Désactivez un démarrage automatique d’un service ?

Pour désactiver le démarrage automatique d’un service, j’ai utilisé la commande :

```
sudo systemctl disable nom_du_service
```

Par exemple, si je souhaite empêcher le service `apache2` de démarrer automatiquement, je ferais :

```
sudo systemctl disable ssh.service
```

Le service reste installé et peut être démarré manuellement si nécessaire, mais il ne s’activera plus au démarrage du serveur.

Conclusion

Ce TP m’a permis de mettre en pratique plusieurs commandes Linux pour le dépannage réseau et la gestion des services. Certaines commandes, je les connaissais déjà parce qu’on les avait vues en cours, et d’autres, je les ai découvertes en cherchant un peu sur Internet, ce qui m’a beaucoup aidé à comprendre comment tout fonctionne.